UMASS Approaches to Detection and Tracking at TDT2

Ron Papka, James Allan, and Victor Lavrenko

Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts
Amherst, MA 01003

ABSTRACT

The following work describes our solutions to the detection and tracking problems defined by the Topic Detection and Tracking (TDT2) research initiative. We discuss the implementation and results of the approaches which were recently tested on the TDT2 evaluation corpus. Our solutions to these problems extend text-based ranked retrieval techniques previously used for document clustering and filtering tasks. We present the effects of different on-line hierarchic clusterings on the detection task. In addition we compare adaptive and static approaches for building linear text classifiers for the tracking task.

1. INTRODUCTION

The motivation for our solutions to the detection and tracking tasks is the hypothesis that several properties of broadcast news can be modeled and incorporated as extensions to a ranked-retrieval engine of Inquery [3]. Given that the information domain for TDT2 is a temporally ordered stream of news, we find that modeling the properties of *time* and *evolution* results in lower story-weighted cost for detection and tracking.

1.1. Implementation

In the experiments that follow, we used the same underlying text representation and threshold model for both problems. Our implementation comprises document and query objects. Documents contain the tokenized text supplied by an automatic speech recognition (ASR) and segmentation system. We create an inverted file from the stemmed terms in each document. Document boundaries were provided by the Linguistic Data Consortium (LDC)[1] for the TDT2 corpora.

Queries are used to represent the contents of a document or a set of documents. A threshold is estimated for each query which determines binary classification decisions. If a new document appearing on the stream has a similarity value exceeding a query's threshold, the document is assumed to discuss the same topic the query attempts to represent. The query and threshold jointly form a classifier which can be stored in an inverted file for on-line applications.

The similarity between a query and document is based on Inquery's belief function. Queries were formulated using the #WSUM operator. A tf representation was used for query weights, and a $tf \cdot idf$ representation was used for document weights, where

$$tf = t/(t + 0.5 + 1.5 * \frac{dl}{avg\_dl}), \text{ and} \qquad (1)$$

[1] http://www.ldc.upenn.edu

$$idf = \frac{log(\frac{C+.5}{df})}{log(C+1)}. \qquad (2)$$

For both query and document term weights, t is the term's frequency in the document, and dl is the document's length. The remaining values for Equations 1 and 2 are derived from an auxiliary corpus. C is the number of documents in the auxiliary corpus, avg_dl is the average number of terms in a document, and df is the number of documents in which the term appears. If the term does not appear in the auxiliary corpus a default value of 1 is used for df.

During processing, a query's threshold is potentially recomputed at each time step, that is, when a new document arrives on the stream. For any query formulated at time i, its threshold for any document arriving at time j is

$$threshold(q_i, d_j) = 0.4 + \theta * (belief(q_i, d_i) - 0.4) + \beta * (date_j - date_i), \qquad (3)$$

where

$$belief(q_i, d_j) = \frac{\sum_{k=1}^{N} q_{i,k} \cdot d_{j,k}}{\sum_{k=1}^{N} q_{i,k}}, \qquad (4)$$

and k is the index of the term weight. The parameters θ and β are determined by an optimization process discussed below. The constant 0.4 is an Inquery belief function parameter, and $(date_j - date_i)$ is the number of days between the arrival of documents d_j and d_i.

1.2. Optimization

During training and development, we honed our systems with the goal of minimizing the TDT2 cost function over all available data. Most of the optimization efforts for detection were based on finding θ and β that minimize story-weighted cost. The optimization efforts for tracking involved finding appropriate values for θ, which we found to be different for different numbers of on-topic training documents (Nt). We determined that thresholds which optimize cost on the training data are consistently higher than optimal thresholds for the test data. We discuss thresholding issues in terms of estimator bias in Section 3.1.

The cost function used for optimization and evaluation in the following experiments is

$$Cost = cost_{fa} * P(fa) + cost_m * P(m) \qquad (5)$$

where $P(fa)$ is the probability a system produces a false alarm i.e., classifying an off-topic document incorrectly, and $P(m)$ is the probability a system produces a miss i.e., classifying an on-topic document incorrectly. In TDT2, cost was defined with $cost_{fa} = 0.98$ and $cost_m = 0.02$

2. DETECTION

One correlation evident in the TDT1 and TDT2 corpora is that news stories appearing on the stream closer in time are more likely to contain discussion of the same topic than stories appearing further apart. We exploited the temporal relationship between stories for the first story detection task of TDT1 by explicitly using a time component in our threshold model [1]. We applied the same threshold model to the TDT2 detection task, and improved the time component by incorporating the number of days between documents, while in TDT1 the time component was based on a document sequence number.

We view the detection task as a problem appropriate for single-pass clustering techniques such as those presented by van Rijsbergen [8]. For the TDT2 detection task we tested single-, average-, and complete-link hierarchic clustering approaches using the $tf \cdot idf$ query and document representations described above.

Our basic algorithm for detection was the following: for each document we formulate a fixed length query from the n most frequent words in the document after removing stopwords. The query's initial threshold is its similarity value to the document from which it was created. We assume no subsequent document will exceed this threshold, and so it is reduced using appropriate values for θ which are discussed below. As new documents arrive on the stream they are compared to previously formulated queries. The existing queries' thresholds are increased based on the new document's relative time of arrival using Equation 3. In the single-link approach, the new document is assigned to the cluster of the query whose threshold it exceeds most. The document starts a new cluster if it does not exceed any existing query's threshold. The average- and complete-link approaches are similar, but require the additional step of assessing the new document's similarity to all the queries in each of the existing clusters. Based on our implementation, we did not find complete-link to be as effective as average-link in terms of TDT2 cost, and therefore we focused on comparing average- to single-link hierarchic approaches. In the following experiments, detection decisions were made in a strict on-line setting which does not use a deferral period (i.e., DEF=0).

2.1. Parameter Estimation for Detection

The single-link implementation ran one order of magnitude faster than the average-link method. Theoretically, both approaches have the same running times. However, average-link requires that each document be compared to all existing queries, while the comparisons for single-link are significantly reduced by using an inverted index file. The faster running time for single-link allowed us to run a semi-exhaustive search for optimal values of θ and β using the data from the TDT1, TDT2 train and development corpora. The parameters that minimized story-weighted cost were applied to both ASR+NWT and CCAP+NWT conditions for the recent evaluation by NIST [2]. The running time of the average-link approach prohibited an extensive parameter search for β, and we were not able to fully test average-link combined with a time component. In the experiments that follow, for the single-link+time approach we set $\theta = 0.22$ and $\beta = 0.001$, for single-link $\theta = 0.3$ and $\beta = 0.0$, and for average-link $\theta = 0.1$ and $\beta = 0.0$.

[2]Both data sets include the same Newswire sources. The ASR+NWT used Dragon Systems Automatic Speech Recognition process applied to Broadcast News sources, while the CCAP+NWT used manually transcribed Closed Caption data for Broadcast News sources.

The number of single term features in the query determines the dimensionality of the underlying vectors used in our implementation. The greater the dimensionality, the slower the running time on sequential machines, so for efficiency, it is preferable to run at lower dimensionality. During the development phase, we varied dimensionality and tested 25, 50, 100, and 200 term queries. We found that at optimal thresholds, lower cost was obtained for some topics using more terms, but for other topics, fewer terms were more effective. In the experiments that follow we used 50 terms, which appeared to work best on the TDT2 train and development corpora for both single- and average-link approaches.

We found that all approaches were affected by the headers and trailers contained in Broadcast News programs. Boundaries provided by LDC indicated that most of these snippets were tagged as miscellaneous text, which are excluded from evaluation, but not from processing. This implies that a system could be indirectly penalized for correctly identifying a topic in a header or trailer by using the terms in the snippet to represent a cluster. For example, a header to ABC's *World News Tonight* may include a brief description of all the topics to be discussed in the program, including one of the target topics being evaluated. We analyzed the length of documents marked miscellaneous in the TDT2 train and development sets, and determined that documents which contain less than 50 terms are very likely to be headers or trailers. We therefore labeled documents under this size as off-topic for all topics.

2.2. Results

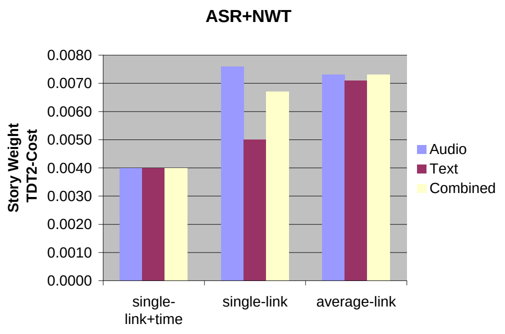

Figure 1: Story-weighted cost for different hierarchic-clustering approaches on ASR and newswire source conditions.

The data from our detection runs are listed in Figures 1 and 2. The charts include separated story-weighted TDT2 cost for audio (ASR or CCAP) and text (NWT). They suggest that single-link clustering incorporating a time component ($\beta > 0$) works as well as, or better than other hierarchic clustering approaches when applied to the detection task for both audio and text.

Single-link+time clustering provided relatively lower story-weighted cost for both ASR+NWT and CCAP+NWT source conditions. The other approaches appeared to benefit from the CCAP

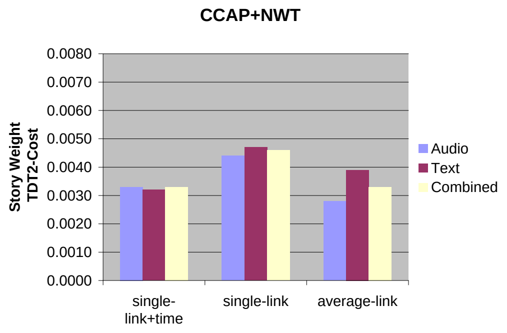

Figure 2: Story-weighted cost for different hierarchic-clustering approaches on Closed Caption and newswire source conditions.

source, from which average-link provided equally low cost to single-link+time. These results corroborate those we obtained using the same systems on the TDT2 train and development corpora. We believe the single-link+time approach works well because it captures the periodicity and overall temporal relationship between news stories.

The single-link+time implementation showed comparable story-weighted cost to the top systems evaluated by NIST for both the ASR+NWT and CCAP+NWT source conditions. Several of these systems used an explicit time component or dynamic-idf. We have observed that dynamic-idf has a similar property to a time component caused by decreasing idf values as more of the stream is processed. It is surprising, however, that single-link approaches worked as well as agglomerative and average-link approaches in light of past observations by Voorhees [9] and Willet [10] both of whom have suggested that average- and complete-link are more effective than single-link for cluster-assisted retrieval.

We analyzed the clusters resulting from each approach. The most salient distinction observed was the number of clusters the different approaches produced. After processing the ASR+NWT stream, which contained 22,445 documents using the boundaries provided, the average-link approach produced 1221 clusters. However, the single-link produce 8206, and single-link+time produced 7515 clusters. Despite similar average cost, the single-link approach suggests that there are nearly 6 times as many topics in the evaluation corpus than the number suggested by the average-link approach.

The representation we used for detection requires the on-line construction of a sparse similarity matrix with $O(n^2)$ entries. Though we were not adversely affected by the space needed by the TDT2 corpora, this requirement is an issue for very large sources. One of the steps that we did not test is agglomeration. An agglomeration step, as discussed by Yang et al. [11], would merge queries in the same cluster into a centroid representation. If agglomeration does not significantly change the number of clusters produced by each approach, then average-link would appear to require less space than single-link using TDT2 cost as a utility function.

More experiments would be necessary to determine which approach is producing the correct granularity of clusters, and those experiments would require exhaustive relevance judgements. However, we were able to test another aspect of clustering using these approaches on the TDT1, TDT2 train and development corpora. The purpose of this analysis was to compare the ability to produce the correct cluster-seed i.e., the first story of a topic. We used software developed for the TDT1 Pilot Study [2] and tested first story detection capabilities for each approach using the parameters for the evaluation runs in TDT2. This retrospective analysis suggested that single-link+time consistently detected 50% more first-stories than average-link at similar false alarm levels.

3. TRACKING

We viewed the tracking task as a text classification problem. Much of the previous work in this area uses a ranked-retrieval engine and determines thresholds at some point in the ranked list of documents. The implementations we tested involved formulating queries and estimating their thresholds assuming an on-line setting. Query formulation involved a three step process:

1. Term Selection.
2. Weight Assignment.
3. Threshold Estimation.

We tested static approaches where query terms and thresholds are held constant over time, and adaptive approaches where documents assumed to be on-topic are used to reformulate the query over time.

Static query formulation served as our baseline system during the training and development stages. Queries are formulated for each topic using the n most frequent nonstopwords from the on-topic documents in the training data. The words are given weights using an assignment based on tf (Equation 1).We also experimented with feature selection and weight assignment variants of the baseline process. We tested static queries expanded with multiword features (MWF) [6]. We also tested two weight-learning algorithms: Dynamic Feedback Optimization (DFO) [7] and Exponentiated Gradient Descent (EG) [5].

3.1. Parameter Estimation for Tracking

We used optimal-parameter searching over the available TDT corpora to determine global threshold parameters. The parameters we found that optimized story-weighted cost on the TDT1, TDT2 train and development corpora are listed in Table 1. In the following section we describe the method we used to obtain these parameters automatically.

Nt	Theta
1	0.2
2	0.3
4	0.4
8	0.6
16	0.8

Table 1: Static tracking threshold parameter θ when $\beta = 0.0$.

The Histogram Method An important component of our tracking system is a threshold estimator for queries formulated for the

Inquery retrieval engine. In Equation 3, we let $\beta = 0.0$, and we use the resulting function as a threshold estimator $\hat{u}$ such that $\hat{u} = 0.4 + \theta(b_{optimized} - 0.4)$, where θ is same global system parameter described above, and $b_{optimized}$ is the similarity value resulting from the query that, when applied to the topic's labeled training documents, optimizes the target cost defined by Equation 5.

Using data from several experiments on the TDT1, TDT2 train and development corpora, it was determined that when fewer on-topic training documents were used, $b_{optimized}$ (our estimator $\hat{u}$ when $\theta = 1.0$) was consistently below the parameter u it was trying to estimate i.e., the optimal threshold for the unprocessed stream of data for a particular topic, or simply $b_{optimal}$. In what follows, the bias in our estimator $\hat{u}$ is the quantity $b(\hat{u}) = E[\hat{u}] - u$.

We observed, as we tested various values for θ, that bias decreased when more on-topic training documents were used to formulate queries. We also observed similar but less significant increases in bias when more features were used. The observation that increasing training instances reduces the bias of an estimator, in general, is not surprising. James, for example, shows that estimates move toward the true population values when training instances are increased for data assumed to have multivariate-normal distributions [4]. He also proposes that once found, it may be possible to reduce the bias using a linear transformation. We tested James's theory using estimator $\hat{u}$. If we define a threshold estimator $\hat{o}$ such that $\hat{o} = b_{optimized}$, then estimator $\hat{u}$ is a linear transformation of estimator $\hat{o}$. Finding optimal values for θ can be done using statistical techniques.

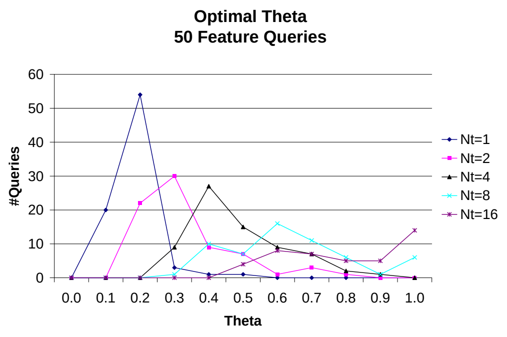

Figure 3: Distributions of optimal threshold parameter θ for varying Nt.

The histogram approach was trained on 91 topics and associated labeled documents from the TDT1, TDT2 train and development corpora. Experiments were conducted over varying numbers of on-topic training documents (Nt) and varying numbers of query/document features. The histogram method, which utilizes threshold estimator $\hat{u}$, is illustrated with 50-feature queries in Figure 3. Histograms of optimal values for θ are collected for each value of Nt and for queries formulated with 10, 20, 50,100, 200, 600, and 10000 features. Using this method on the training data determined that for 1 on-topic document ($Nt = 1$) and 50-feature queries, 54 out of 91 queries had optimal cost when $\theta = 0.2$. For each pair of Nt and number of features, we calculate $E[\theta]$ from the corresponding histogram, and used the nearest tested value for estimator $\hat{u}$ when conducting evaluation experiments using the same pair. From the data in Figure 3, it is evident that as Nt increases $E[\theta]$ increases. This observation coupled with the definition of estimator $\hat{u}$ implies that $b_{optimized}$ is closer to $b_{optimal}$ when more on-topic training examples are used.

Varying IDF source Using static queries as a baseline approach, we evaluated different sources for IDF over varying dimensionality for overall minimum cost on the TDT1, TDT2 train and development corpora. We found that *idf* calculated from document frequencies (*df*) from TREC volumes 1,2,3 combined with TREC-4 Routing (TREC 123R) produced lower cost and DET curves than *idf* from the news-only documents from TREC 123R and 80K documents from a CNN corpus. Also, TREC 123R was a more effective source for document frequency than the retrospective corpus being processed. In general, the overall cost changes between using different sources for *idf* and not using *idf* at all were small at best. Our experience using dynamic *idf* from TDT1 tracking, however, suggests that document frequency is a useful statistic. In addition, we found that using queries comprised of 50 terms was more effective than using queries containing up to 10, 20, 100, 200, 600, and 10000 terms.

Adaptive Query Formulation In addition to our static approach we tested adaptive tracking approaches that extend some techniques previously used to analyze the TDT1 corpus [1]. In the adaptive approach, a query is initially formulated using the static approach, and then reformulated on-line with features from new documents in the stream. Both on-topic and off-topic documents are saved and used to reformulate a query.

The threshold model for adaptive tracking uses Equation 3 for two thresholds. One threshold is estimated for detection decisions and another threshold is estimated for agglomeration decisions. If a document exceeds the second threshold, it is used to reformulate the query and determine new thresholds. If the document similarity to the query was above the decision threshold but below the agglomeration threshold, then it was excluded from the reformulation process.

After reformulating a query, we backtested its training data and calculated TDT2 cost. If cost increased, we discarded the reformulated query and reverted to the previous one. This additional step was necessary to prevent the query from over-generalizing. In addition, we found that reformulating queries with 10 terms worked better than 20 and 50 terms.

In the adaptive tracking experiments that follow, we determined values for θ for different values of Nt using an optimization process over the TDT1, TDT2 train and development corpora. As with the static approach we let $\beta = 0.0$, which turns off the time component of the threshold equation. We found that an agglomeration threshold when $\theta = 0.85$ worked best. The decision thresholds that we found from the optimization process appear in Table 2.

Nt	Theta
1	0.45
2	0.55
4	0.65

Table 2: Adaptive tracking threshold parameter θ with $\beta = 0.0$.

Decision Score Normalization The DET curve illustrates the trade-off between system miss and false alarm rates. Since Equations 1-4 produce different distributions of values for different topics, we needed to normalize the similarity values from Equation 4 to produce decision scores. The TDT2 decision score is defined so that all on-topic decisions must have a higher value than off-topic decisions. Our initial attempt at normalization did not meet this requirement, but in what follows, we produce decision scores based on several experiments done after the submission deadline. Hard decisions were made using the same process described above and match the submitted runs.

In these experiments we looked at different approaches to belief value normalization and their ability to lower DET curves on the DET graph. We found that the most effective way to lower curves was to normalize similarity score using a standard normal transformation. The general form of the transformation is

$$decision\_score(q_i, d_j) = \frac{belief(q_i, d_j) - \mu}{\sigma}, \qquad (6)$$

where μ is the mean, and σ is the standard deviation of some distribution that is *assumed* to be normal. We observed that queries for most topics produced distributions of similarity values that were relatively normal for on-topic documents, but the distributions did not appear normal for off-topic documents. Experiments setting μ and σ to the values estimated from the on-topic documents in the training sample produced worse DET curves than using the estimates calculated from the off-topic documents. In either case, this transformation did not satisfy the decision score requirement that on-topic decisions have a higher score than off-topic decisions.

We found that setting $\mu = threshold(q_i, d_j)$, and σ to the standard deviation of similarity values in the training data resulted in good curves for both static and adaptive approaches. This normalization appeared to work as well as using the distributions from the off-topic training documents, with the additional property of producing decision scores normalized around a value of 0. We used this normalization process for the DET curves produced below.

3.2. Results: Static vs. Adaptive Tracking

Type	Nt	ASR+NWT	CCAP+NWT
Static	4	0.0070	0.0066
Static	2	0.0069	0.0071
Static	1	0.0073	0.0081
Adaptive	4	0.0059	0.0064
Adaptive	2	0.0107	0.0075
Adaptive	1	0.0103	0.0122

Table 3: Story-weighted cost for static and adaptive tracking.

A comparison of our static and adaptive tracking approaches is listed in Table 3. The data indicate that adaptive queries were more effective than static queries in terms of story-weighted average cost at $Nt = 4$. The adaptive approach had lower cost for 9 topics and higher cost for 7 topics. However, the results in Table 3 and Figure 4 suggest that the adaptive approach is less effective at lower values of Nt.

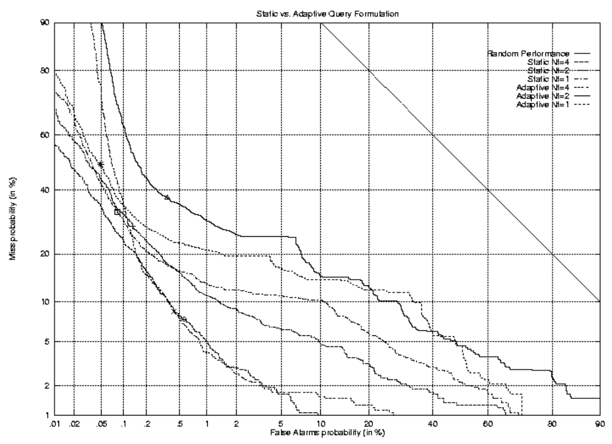

Figure 4: DET graph: static vs. adaptive tracking (ASR+NWT).

The DET graph in Figure 4 illustrates that at $Nt = 4$ the adaptive curve is closer to the origin than the static query curve for $P(m) < 20\%$; however, the static curve has a more desirable error detection trade-off. The static queries resulted in lower DET curves than their adaptive counterparts for $Nt = 2$ and $Nt = 1$. The DET graph suggests that the static query process is more robust than the adaptive approach. With the exception of adaptive tracking at $Nt = 2$, both approaches showed no improvement on the CCAP+NWT source. This suggests that the impact of ASR on tracking was minimal.

3.3. Results: Multiword Features and Weight-Learning

We tested various extensions of static query formulation that have been hypothesized to significantly improve retrieval effectiveness over basic Rocchio methods. Multiword features (MWF) were found to improve tracking effectiveness in subsets of the training corpora. We used a process described by Papka and Allan [6] that expanded queries with Inquery proximity operators of varying size windows of words. We tested windows of 5, 20, 50, and 100 words.

The weight-learning algorithms adjust the static query's term weights using supervised training techniques. The DFO algorithm was the same as that used by Schapire et al. [7]. This algorithm tweaks each query weight in turn, and recomputes average precision on the training data. If a weight change does not improve precision, the change is undone. The EG algorithm was a modification of the algorithm used by Lewis et al. [5]. This learning technique is similar to other least-square-error reduction approaches. Query weights are adjusted based on pre-specified target values for on- and off-topic documents. The algorithm attempts to minimize the difference between the target values and the actual belief values produced by the training data.

In the following experiments, a static query was generated using the training data containing $Nt = 4$ on-topic instances, and the off-topic documents supplied by NIST. Before tracking begins, the static query is expanded using multiword features, or weights are modified using EG or DFO. We used the same threshold parameters as those in Table 1. The results are listed in Table 4.

Type	Story-Weighted	Topic-Weighted
Static	0.0070	0.0066
Adaptive	0.0059	0.0074
Static+DFO	0.0061	0.0067
Static+EG	0.0070	0.0080
Static+MWF	0.0072	0.0064

Table 4: TDT2 cost for extensions to static tracking. (ASR+NWT, Nt=4).

As with the adaptive approach, we found that expansion and weight-learning approaches did not improve overall effectiveness significantly. For the 21 topics evaluated, improvements were minimal. The queries expanded with multiword features had lower cost for 10 topics, but increased cost for 6. Further analysis of the weight-learning approaches revealed that the DFO algorithm decreased cost for 3 topics, and increasing cost for 2 topics. EG decreased cost for 1 topic, and increased cost for 1 topic.

The weight-learning approaches are of little use at $Nt \leq 4$ using our representation. We find that most queries and thresholds already separated the training data, so not much improvement should be expected from weight-learning [3]. We observed that for higher values of Nt the occurrence of training data separation decreases. An analysis of the TDT2 train and development corpora, for example, revealed that 95% of the queries and thresholds formulated with 4 on-topic training documents separated the training data. At $Nt = 16$ only 10% of the classifiers separate their training data. This suggests that weight-learning algorithms are more likely to be effective for higher values of Nt, and training data separation should be tested at lower values.

4. Conclusion

We presented several solutions to the Detection and Tracking problems defined by the Topic Detection and Tracking research initiative. For detection, we compared single-pass hierarchic clustering solutions including single-link and average-link approaches. The data suggest that augmenting single-link clustering with a time component (single-link+time) yields low cost on the target cost function and source conditions. Other detection experiments using our representation suggested that average-link is comparable to single-link+time when audio sources for news are manually transcribed. We suggest using single-link+time for on-line clustering of Broadcast News because it is faster than average-link. Furthermore, we find that single-link+time is more effective for on-line new event detection (first story detection) than average-link, which suggests that the clustering granularity produced by single-link+time is closer to the actual granularity of *topic* defined by the relevance assessments.

We viewed the tracking problem as an instance of on-line document classification, and used an extension of techniques that have been shown to work well for similar problems such as document filtering. We compared variations of a static query formulation process that included query expansion with multiword features and weight-learning steps. In addition, we tested adaptive query formulation, which has the side-effect of including new features in the query over time. We found insignificant effectiveness improvements using these variations, which in most cases require significant computing resources. The adaptive technique appeared to work well on the target evaluation condition ($Nt = 4$), but proved less robust than the static approach for $Nt < 4$. The weight adjustment algorithms affected very few topics at $Nt \leq 4$. For higher values of Nt, however, we expected significant improvements in tracking effectiveness using these query expansion and weight-learning approaches.

[3] When the training data are separated it implies that a query and threshold result in a TDT2 cost of 0. Furthermore, no further improvements to Average Precision (used by DFO) can be obtained because the training data is perfectly sorted.

5. Acknowledgments

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

1. J. Allan, R. Papka, V. Lavrenko, "On-line New Event Detection and Tracking," *Proceedings of ACM SIGIR*, pp.37-45, 1998.
2. J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic Detection and Tracking Pilot Study: Final Report," *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, 1998.
3. J. Callan, B. Croft, and J. Broglio, "TREC and TIPSTER Experiments with INQUERY," *Information Processing & Management*, 31(3):327-343, 1994.
4. M. James, *Classification Algorithms*, John Wiley & Sons, New York, 1985.
5. D. Lewis, R. Schapire, J. Callan, and R. Papka ,"Training Algorithms for Linear Text Classifiers," *Proceedings of ACM SIGIR*, pp.298-306, 1996.
6. R. Papka and J. Allan, "Document Classification using Multiword Features," *Proceedings of ACM CIKM*, pp.124-131, 1998.
7. R. Schapire, Y. Singer, and A. Singal ,"Boosting and Rocchio Applied to Text Filtering," *Proceedings of ACM SIGIR*, pp.215-223, 1998.
8. C.J. van Rijsbergen, *Information Retrieval, 2ed.*, Butterworths, Massachusetts, 1979.
9. E. Voorhees, *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval* , Ph.D. Thesis, 1985.
10. P. Willet, "Recent Trends in Hierarchic Document Clustering: A Critical Review," *Information Processing & Management*, 24(5):577-597, 1988.
11. Y. Yang, T. Pierce, and J. Carbonell, "A Study on Retrospective and On-Line Event Detection," *Proceedings of ACM SIGIR*, pp.28-36, 1998.